Setting up a New MOOC Front-End

Rich Neswold
Fermi National Accelerator Laboratory
Beams Division/Controls Department

PO Box 500, MS 360
Batavia, IL 60510

voice: (630) 840-3454
fax: (630) 840-3093

neswold@fnal.gov

$Date: 2001/10/26 $

This document details the steps needed to create a MOOC-based front-end. Once these steps have been completed, the reader should have a front-end that supports ACNET protocols. This document doesn't describe MOOC driver development.

1. Introduction

The job of a front-end is to bridge data acquisition hardware with the ACNET protocols. A piece of hardware, controlling a piece of the accelerator, sits on a *field bus*. The field bus could be an RS-232C connection, or an ARCnet LAN, or can even be simply the local VME backplane. The front-end accepts ACNET requests and forwards them to the devices.

MOOC front-ends are currently 68000- or PowerPC-based systems that reside in a VME crate. These systems run the VxWorks™ RTOS from Wind River Systems (http://www.wrs.com).

2. Building a System

This section describes the steps and details one needs to follow in preparing a system to run MOOC.

2.1. Register Network Addresses

2.1.1. IP Address

Preparing a processor card for VxWorks™ requires network access. While you are obtaining the hardware, the network group can be preparing the system's network presence (this step may have to be delayed until you know the MAC address of the processor card.) You can apply for a new IP address on the web using the new node form (http://www-bdnew.fnal.gov/network/nodeform.html).

Once the request is completed, your node name will get propagated to the name servers over the next several days.

2.1.2. ACNET Address

Pick out a node name (6 characters, or less) for your system. Give this name, along with the IP name of your system to Brian Hendricks (<`hendricks@fnal.gov`>). He will add your node to the ACNET node tables. Your node will also be assigned a *trunk* and *node*, which are ACNET addressing information. You'll need to remember these values when setting up the boot parameters.

2.2. Configuring the Hardware

You'll need a VME processor card and a VME crate to house it. The supported VME cards are listed in Table 1. You'll also have to obtain a VxWorks™ run-time license for your front-end. You can see the Front-End Group in the Controls Department for details on obtaining a license.

Table 1. Supported VME processors

Processor	Description
MVME-162	MC68040 VME card with 4 or 16 MB of RAM
MVME-2300	PowerPC 603 VME card with 16 MB of RAM
MVME-2400	PowerPC 750 VME card with 32 MB of RAM

We have licensed *board support package*s from Wind Rivers for these processor cards. If you want to use a different card, you'll have to obtain your own BSP. Also, if your card doesn't contain one of the processors, you'll have to get a license for the compiler tools. The moral is to use one of the listed processor cards.

2.2.1. Installing a Boot Image

The processor cards come from the factory with a monitor/debugger in its ROM. Wind Rivers provides a boot image that is tailored for the VxWorks™ RTOS. Since we typically don't use the built-in debugger, we replace the boot code with the Wind River version. Each processor card has a slightly different way of installing this image.

2.2.1.1. MVME-162

Installing a boot image requires that a new ROM be installed. We have copies of the ROM available, so if you need one, you can contact me. For these processors, the Motorola ROM is removed and the VxWorks™ boot ROM is installed in its place.

2.2.1.2. MVME-230x Series

For this series of cards, the boot image is placed in the onboard flash memory. Make sure you have the system jumpers set up to boot the system out of bank *B* of the flash memory. This is done by moving J15 to pins 2 and 3. Boot the processor card so that you're in the **PPCBUG** monitor. You'll see something like this:

```
Copyright Motorola Inc. 1988 - 1997, All Rights Reserved

PPC1 Debugger/Diagnostics Release Version 3.5 - 01/30/98 RM01
COLD Start

Local Memory Found =01000000 (&16777216)

MPU Clock Speed =200Mhz

BUS Clock Speed =67Mhz

Reset Vector Location  : ROM Bank B
Mezzanine Configuration: Single-MPU
Current 60X-Bus Master : MPU0
Idle MPU(s)            : NONE

System Memory: 16MB, ECC Enabled (ECC-Memory Detected)
L2Cache:       NONE

PPC1-Bug>
```

Enter your IP address using the `niot` command. An example of this is given for `iowa.fnal.gov`, a test front-end:

```
PPC1-Bug> niot
Controller LUN =00?
Device LUN     =00?
Node Control Memory Address =00F9E000?
Client IP Address      =0.0.0.0? 131.225.123.64
Server IP Address      =0.0.0.0? 131.225.121.207
Subnet IP Address Mask =255.255.0.0? 255.255.255.0
Broadcast IP Address   =255.255.255.255? 131.225.123.255
Gateway IP Address     =0.0.0.0? 131.225.123.200
Boot File Name ("NULL" for None)     =?
Argument File Name ("NULL" for None) =?
Boot File Load Address         =001F0000?
Boot File Execution Address    =001F0000?
Boot File Execution Delay      =00000000?
Boot File Length               =00000000?
Boot File Byte Offset          =00000000?
BOOTP/RARP Request Retry       =00?
TFTP/ARP Request Retry         =00?
Trace Character Buffer Address =00000000?
BOOTP/RARP Request Control: Always/When-Needed (A/W)=W?
BOOTP/RARP Reply Update Control: Yes/No (Y/N)       =Y?

Update Non-Volatile RAM (Y/N)? y
```

You can verify your configuration by using the `niot;h` command.

```
PPC1-Bug> niot;h
Network Controllers/Nodes Available
CLUN DLUN Name     Address    P-Address/H-Address
   0    0 DEC21140 $80804000  131.225.123.64/08003E25C06E
PPC1-Bug>
```

Now that the network interface is configured, it is time to download the boot image into RAM. The boot image is downloaded via the **tftp** protocol. The server is `nova.fnal.gov`. Use the `niop` command to download the file.

```
PPC1-Bug> niop
Controller LUN =00?
Device LUN     =00?
Get/Put        =G?
File Name      =? vxworks/bootroms/mv2303_5.4.bin

Memory Address =00004000? 100100
Length         =00000000?
Byte Offset    =00000000?

Bytes Received =&434952, Bytes Loaded =&434952
Bytes/Second   =&217476, Elapsed Time =2 Second(s)

PPC1-Bug>
```

Now that the file has been downloaded, it is time to program the flash.

```
PPC1-Bug> pflash 100000 1fffff ff000000;r
Source Starting/Ending Addresses      =00100000/001FFFFF
Destination Starting/Ending Addresses =FF000000/FF0FFFFF
Number of Effective Bytes             =00100000 (&1048576)

Program FLASH Memory (Y/N)? y
```

Once the flash has been programmed, the system will automatically reboot. Once the system has reached the **PPCBUG** prompt, remove power from the system and replace the jumper so that the system boots out of the new image.

2.2.1.3. MVME-240x Series

For this series of cards, the boot image is placed in the onboard flash memory. Make sure you have the system jumpers set up to boot the system out of bank *B* of the flash memory. This is done by moving J8 to pins 2 and 3. Boot the processor card so that you're in the **PPCBUG** monitor. You'll see something like this:

```
Copyright Motorola Inc. 1988 - 1998, All Rights Reserved

PPC4 Debugger/Diagnostics Release Version 1.1 - 01/11/99 RM01
COLD Start

Local Memory Found =02000000 (&33554432)

MPU Clock Speed =233Mhz
```

```
BUS Clock Speed =67Mhz

Reset Vector Location  : ROM Bank B
Mezzanine Configuration: Single-MPU
Current 60X-Bus Master : MPU0
Idle MPU(s)            : NONE

System Memory: 32MB, ECC Enabled (ECC-Memory Detected)
L2Cache:       1024KB, 93Mhz

PPC4-Bug>
```

Before the network interface can be used, the real-time clock must be enabled. At this point you should set up the clock (you'll have to replace the *MMDDYYHHMM* with the current time.)

```
PPC4-Bug> set MMDDYYHHMM
PPC4-Bug>
```

Enter your IP address using the `niot` command. An example of this is given for `iowa.fnal.gov`, a test front-end:

```
PPC4-Bug> niot
Controller LUN =00?
Device LUN     =00?
Node Control Memory Address =01F9E000?
Client IP Address      =0.0.0.0? 131.225.123.64
Server IP Address      =0.0.0.0? 131.225.121.207
Subnet IP Address Mask =255.255.0.0? 255.255.255.0
Broadcast IP Address   =255.255.255.255? 131.225.123.255
Gateway IP Address     =0.0.0.0? 131.225.123.200
Boot File Name ("NULL" for None)     =?
Argument File Name ("NULL" for None) =?
Boot File Load Address         =001F0000?
Boot File Execution Address    =001F0000?
Boot File Execution Delay      =00000000?
Boot File Length               =00000000?
Boot File Byte Offset          =00000000?
BOOTP/RARP Request Retry       =00?
TFTP/ARP Request Retry         =00?
Trace Character Buffer Address =00000000?
BOOTP/RARP Request Control: Always/When-Needed (A/W)=W?
BOOTP/RARP Reply Update Control: Yes/No (Y/N)       =Y?

Update Non-Volatile RAM (Y/N)? y
```

You can verify your configuration by using the `niot;h` command.

```
PPC4-Bug> niot;h
Network Controllers/Nodes Available
CLUN DLUN Name     Address    P-Address/H-Address
   0    0 DEC21140 $80804000  131.225.123.64/08003E25C06E
PPC4-Bug>
```

Now that the network interface is configured, it is time to download the boot image into RAM. The boot image is downloaded via the **tftp** protocol. The server is `nova.fnal.gov`. Use the `niop` command to download the file.

```
PPC4-Bug> niop
Controller LUN =00?
Device LUN     =00?
Get/Put        =G?
File Name      =? vxworks/bootroms/mv2400_5.4.bin

Memory Address =00004000? 100100
Length         =00000000?
Byte Offset    =00000000?

Bytes Received =&427244, Bytes Loaded =&427244
Bytes/Second   =&213622, Elapsed Time =2 Second(s)

PPC4-Bug>
```

Now that the file has been downloaded, it is time to program the flash.

```
PPC4-Bug> pflash 100000 1fffff ff000000;r
Source Starting/Ending Addresses      =00100000/001FFFFF
Destination Starting/Ending Addresses =FF000000/FF0FFFFF
Number of Effective Bytes             =00100000 (&1048576)

Program FLASH Memory (Y/N)? y

Virtual-Device-Number   =00
Manufacturer-Identifier =10001
Device-Identifier       =22C422C4
Virtual-Device-Number   =01
Manufacturer-Identifier =10001
Device-Identifier       =22C422C4
Address-Mask            =FF800000
Erasing sector     =$FF000000
Erasing sector     =$FF040000
Erasing sector     =$FF080000
Erasing sector     =$FF0C0000
Programming sector =$FF000000
Programming sector =$FF040000
Programming sector =$FF080000
Programming sector =$FF0C0000
```

Once the flash has been programmed, the system will automatically reboot. Once the system has reached the **PPCBUG** prompt, remove power from the system and replace the jumper so that the system boots out of the new image.

2.2.2. Setting up Boot Parameters

When a processor is booted, it executes the VxWorks™ loader. The loader is responsible for pulling a kernel across the network. In order to do this, it needs some network information.

Boot the system. You will see some system information and then a countdown will begin. Press the space bar to stop the countdown. You should now see the loader prompt:

```
                              VxWorks System Boot


    Copyright 1984-1998  Wind River Systems, Inc.






    CPU: Motorola MVME2401-1 - MPC 750
    Version: 5.4.2
    BSP version: 1.2/0
    Creation date: Oct 23 2001, 16:42:58





    Press any key to stop auto-boot...
     1 Press spacebar
    [VxWorks Boot]:
```

To change the boot parameters, enter the `c` at the loader prompt. The following example shows how `moosehead.fnal.gov` was set up.

```
    [VxWorks Boot]: c

    '.' = clear field;  '-' = go to previous field;  ^D = quit

    boot device          : dc0
    processor number     : 0
    host name            : host fecode-bd
    file name            : vxWorks vxworks_boot/kernel/mv2400/vxWorks
    inet on ethernet (e) : 10.0.0.1 131.225.123.178:ffffff00
    inet on backplane (b):
    host inet (h)        : 10.0.0.2 131.225.121.145
    gateway inet (g)     :  131.225.123.200
    user (u)             : vxworks vxworks_boot
    ftp password (pw) (blank = use rsh):
    flags (f)            : 0x0 0x8
    target name (tn)     : target nnfern_0x0A3E
    startup script (s)   :
    other (o)            :

    [VxWorks Boot]:
```

The fields probably need some explaining:

boot device. This is the network interface that will be used to load the kernel. The actual device name will vary between each BSP. In the example, a PowerPC board was used so the device name is `dc0`. A 68k system would have a `ln0` for the boot device.

host name. This field holds the name of the system that holds the kernels. The VxWorks™ node will try to download its kernel from this machine.

file name. This is the name of the file that holds the VxWorks™ kernel image. The path names are relative to the login directory (the account name, password and protocol are specified in other fields.)

inet on ethernet. This is the system's IP address. In the example, I added a 24-bit netmask specification.[1] If the netmask (the portion after and including the colon) isn't provided, the kernel will use a 16-bit mask.

host inet. This is the IP address of the machine specified in the *`host name`* field. Since the VxWorks™ kernel doesn't have DNS support, both the name and IP address of the server need to be provided. The kernel will use both pieces of information to make a default entry in the host table.

gateway inet. This is the IP address of the router for the local subnet.

user. This is the account used to download kernels and run-time modules. The server only allows the `vxworks_boot` account access to remote shell service, so all front-ends should be using this account.

flags. This is a set of flags that define some boot options. The `?` command shows the various flags available. The machine used in the example defined the flags as shown in Table 2.

target name. This sets the host name for the VxWorks™ node. A task can retrieve this value by calling `gethostname()`.

ACNET node follow a naming convention. The name of the machine is first, followed by an underscore character, followed by the hexadecimal representation of the ACNET trunk and node. The ACNET libraries use the trunk and node information in the hostname to configure itself.

startup script. This is a file containing VxWorks™ commands. It gets loaded after the kernel loads. In the previous example, no script was specified so the system will simply boot to the VxWorks™ shell prompt.

Table 2. Boot Flags

Value	Description
0x02	load local system symbols
0x04	don't autoboot
0x08	quick autoboot (no countdown)
0x20	disable login security
0x40	use bootp to get boot parameters
0x80	use tftp to get boot image
0x100	use proxy arp

Glossary

board support package

The board support package is the set of drivers, documentation, header files, and libraries that are particular to a given processor board. The VxWorks™ development environment (Tornado) can build applications and kernels. To build a working kernel, however, some board-specific modules are needed. That’s where the board support package fits in.

field bus

The field bus is the connection between a controller and the hardware it controls. The controlled hardware is typically not local to the controller. For example, although the PCI bus is technically a field bus, we wouldn’t refer to it as such since it resides completely within the front-end packaging.

At Fermilab, we use the following field busses: CAMAC serial links, ARCnet LANs, RS-232C, and GPIB.

node

The secondary portion of ACNET addressing information. There can be up to 256 nodes on an ACNET trunk.

trunk

The primary portion of ACNET addressing information. Traditionally, the trunk would indicate the routing information. For instance, front-ends that had token ring interfaces would reside on trunk 0. Since token ring nodes have been replaced by ethernet speaking systems, the trunk has lost a lot of its significance.

Notes

1. The network group has requested that 24-bit net masks be used on the front-ends. The reasons are beyond the scope of this article.